Karachi Koalas3D Simulation Soccer Team
Team Description Paper for World RoboCup 2011

Mary-Anne Williams[1], Sajjad Haider[2], Saleha Raza[2], Benjamin Johnston[1], Osama Khan[2], Shaukat Abidi[2] and Arsalan Ansari[2]

[1]Innovation and Enterprise Research Lab, University of Technology, Sydney, Australia
[2]Artificial Intelligence Lab, Institute of Business Administration, Karachi, Pakistan
http://www.karachikoalas.org
mary-anne.williams@uts.edu.au, sajjad.haider@khi.iba.edu.pk

Abstract. This paper describes the algorithms and techniques developed by Karachi Koalas as it aims to participate in the 3D simulation league. We have developed a partial Fourier series based bipedal gait that is optimized through evolutionary algorithms while localization is accomplished through particle filters. Localization and ball tracking are further enhanced via the message passing mechanism available within the RoboCup 3D Server environment. The strategy code is based on dynamic role switching and fuzzy rules. Currently we are working on case-based and inductive reasoning to further enhance our team strategy.

1 Introduction

Karachi Koalas team was formed in the mid of 2010 as a result of a strong and further evolving scientific partnership between University of Technology, Sydney (UTS) and Institute of Business Administration, Karachi (IBA). UTS has a strong commitment to the RoboCup competition and has been a frequent participant in the Standard Platform League starting from 2003. It won the Australian RoboCup Championship competition in 2004 and was the top International Team in 2004 at Robot Soccer World Cup where it came first in the Soccer Challenges and second in the Soccer Games. Since 2008, it has formed a joint Standard Platform League team, named WrightEagleUnleashed [1], with University of Science and Technology China, which was the Runner-Up of 2008.Several papers have been published by the team members on RoboCup related research topics that demonstrate its commitment and contribution to the advancement of RoboCup [2-9]. IBA, on the other hand, being one of the premier higher education schools in Pakistan, is also committed to robotics related education and has taken several pioneering steps to introduce robotics at high school, undergraduate and graduate levels. The human resource and skill sets present at both UTS and IBA complement each other's research interest perfectly and is already resulting in collaboration in many areas within the realm of RoboCup.

Being a new team in 3D simulation league, we had to write everything from scratch, be it locomotion, localization or team behavior. The rest of the paper describes the development environment and code architecture of Karachi Koalas, advancements made in locomotion, localization and team behavior and the further high priority tasks we aim to finish before the competition.

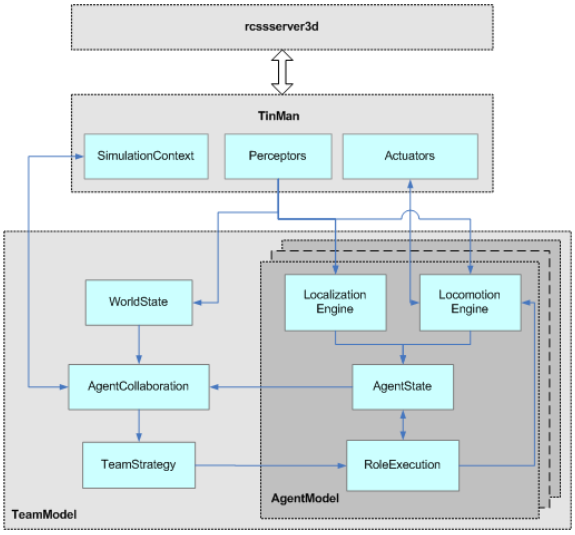


Fig. 1. Software Architecture

2 Development Environment

We are the first team in 3D simulation league which has done all the development in C#/Mono. We've used the newly released TinMan library [10] for communication with the server. In the beginning we also explored littlebats[11] and zigorat[12] libraries in addition to TinMan but then decided to focus on TinMan as it is based on .NET which matches with the development expertise of the KK team members. TinMan's execution on Linux has been made possible through Mono[1]. This flexibility provided us an ideal platform to build our code simultaneously in Windows and Linux environments. We have also made extensive use of the recently released debugging tool RoboViz [13]. The tool is great for the dynamic placement of ball and agents as well as getting insight of agents' internal states and beliefs

3 Software Architecture

We have developed a modular architecture that is built on top of the TinMan library. Fig. 1 provides a high level view of the overall software architecture. TinMan supports low-level interfacing with the RoboCup server (rcsserver3d) by providing higher level abstraction of preceptors and actuators for communication with the server. These actuators and preceptors are

[1] Mono is an open source implementation of .NET framework which enables .NET applications to be developed and executed on Linux.

used by our *AgentModel* and *TeamModel*. *AgentModel* is responsible to handle the functioning of an individual agent. This includes maintaining the current state of the agent in *AgentState*, localizing it in the field using *localization engine* and enabling it to exhibit different types of motion via *locomotion engine*. Locomotion and localization are two key components of *AgentModel* and have been described in detail in the following sections. Overall coordination among agents is performed by *AgentCollaboration* module that gathers an agent's state from *AgentState* and game/world state from *WorldState* and applies different heuristics to devise a suitable strategy. *TeamStrategy* module deals with the execution of a certain strategy by adopting a suitable formation and dynamically assigning different roles to each player. Agents are then responsible to enact these roles using *RoleExecution*. The RoboCup 3D Server supports direct communication among agents through its messaging interface. This interface has also been exploited by the *AgentCollaboration* module that in turn uses *SimulationContext* of TinMan to receive and broadcast messages

4 Locomotion

Our locomotion efforts were focused on the development of the following skill sets:

- Forward, backward, turn and side walks
- Getup from back and belly and diving behavior of the goal keeper
- Forward, side and angular kicks

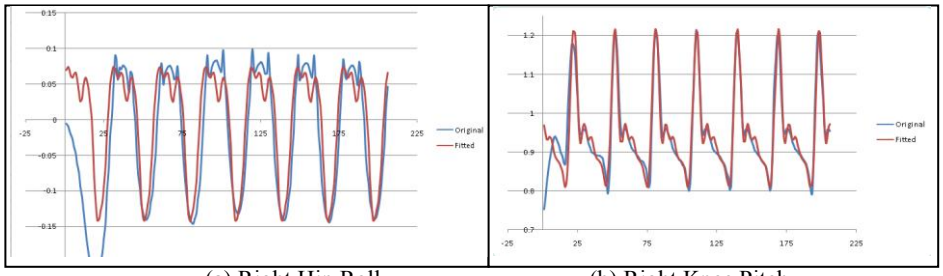
4.1 Forward and Side Walk

For forward, backward, turn and side walk routines, we opted for a computational intelligence/machine learning based approach to learn the periodic motion of relevant shoulder and leg joints. The movement of each joint is modeled using Partial Fourier Series (PFS) of the form

$$f(t) = a_0 + \sum_{n=1}^{N} a_n \sin\left(\frac{2\pi nt}{L} + \phi_n\right)$$

where N is the number of frequencies, a_0 is the offset, $a_n$represents amplitudes, L is the period and φ_n represents phases.

Evolutionary Algorithms were used extensively in this walk learning and optimization process. They were used in two different scenarios: (a) to learn the walk of an actual Nao and (a) to further optimize that walk within the simulation environment. At UTS, we have access to actual Naos and we used them to generate walk data. The default NaoQi walk engine was used to get angles of different joints as Nao performed different walks. Once data was collected for a particular walk, an evolutionary algorithm was used to find parameters of PFS that best fit the data. The process was repeated for each joint. Fig.2 shows the plot of graphs obtained through actual forward walk and the learned PFS for hip roll and ankle pitch movements.



(a) Right Hip Roll (b) Right Knee Pitch

Fig. 2. Graphs of Original Data and Fitted Equations based on Partial Fourier Series

The PFS equations learned through this process had varied degree of complexity. Some of the learned equations were quite simple (based on one frequency) and had only 4 parameters. Quite a few, however, were based on 6 frequencies and as a result had 19 parameters. Overall, the set of equations which involved shoulder pitch, hip yaw, hip pitch, hip roll, knee pitch, ankle pitch and ankle roll for both left and right arms/legs had 96 parameters.

These parameters and the corresponding sets of equations were further evolved through another class of evolutionary algorithms within the simulation environment of RoboCup 3D Server. The evolutionary algorithm was initialized with the mutated copies of the learned parameters. The algorithm ran over several days and the preliminary and intermediate results suggested that a strong harmony existed between the pitch movements (belonging to hip, knee and ankle of both legs), which resulted in the reduction of required parameters. In addition, due to symmetry that exists between left leg and right leg movement, further parameters were reduced. Furthermore, it was found that few of the leg joints do not oscillate much during certain walks and thus do not need to be modeled as PFS. Additionally, it was observed that the period of all PFS should be kept the same to synchronize the joints movements by a single frequency clock. Thus using these insights and others, the parameters were reduced from 96 to 19 (and even lesser for turn) as the algorithm evolved towards a better solution over the course of more than two weeks.

During this parameter searching and optimization exercise, the evolutionary algorithm was guided by a composite fitness function that consists of the following characteristics:

- Distance traveled by the simulated Nao
- Straightness of the walk
- Stability of the walk computed through gyroscope reading

These criteria were used for forward, backward and side walk routines. For turn routine, the top two criteria were altered and instead of the distance travel and straightness of the walk metrics, the focus was on finding a solution which sees most number of unique markers during the allotted time. In all evolutions, solutions which caused the robot to fall on the ground were penalized heavily and similarly with walks/solutions which rotated/drifted excessively. The stability was measured as the average standard deviation of gyro readings on all axes.

$$\text{stability} = \sqrt{\frac{\sum_{i=1}^{N}(x_i-\bar{x})+\sum_{i=1}^{N}(y_i-\bar{y})+\ \sum_{i=1}^{N}(z_i-\bar{z})}{N}}$$

where N represents the number of observations taken during the experiment, x_i , y_i and z_i are the values received from the gyroscope in the i^{th} cycle and $\bar{x}$, $\bar{y}$ and $\bar{z}$are the arithmetic mean of gyroscope readings over the course of experiment.

As the evolutionary algorithms proceeded over a period of more than two weeks, each of the above objectives were given different priority/weights to better guide the search process; but eventually the focus was on the maximum distance traveled by the simulated Nao (in case of forward, backward and side walks) provided that it did not drift too much from a straight/side walk and the stability remained within a certain threshold. This approach to walk optimization using a combination of evolutionary computation techniques and PFS has been suggested by few researchers in the literature. [14-16]

4.1 Getup from Back and Belly

The accelerometer sensor in Nao gives values of gravity for the x,y and z axes. Using these values one can discover the orientation of the robot - whether it is standing up, has fallen on its sides or its back/belly. We have, thus, used these accelerometer values for the identification of the state of the robot: standing or fallen on the ground. Once the robot figures out that it has fallen on the ground, it calls the standup routine. The standup behavior implementation involves key-framed joint movements for each action involved in getting up to a stable position from all possible orientations. A different set of key-frame motions have been defined for *Get up from Belly* in contrast to *Get up from Back*. A similar set of key-framed joint movements have been implemented for goal keeper's diving behavior.

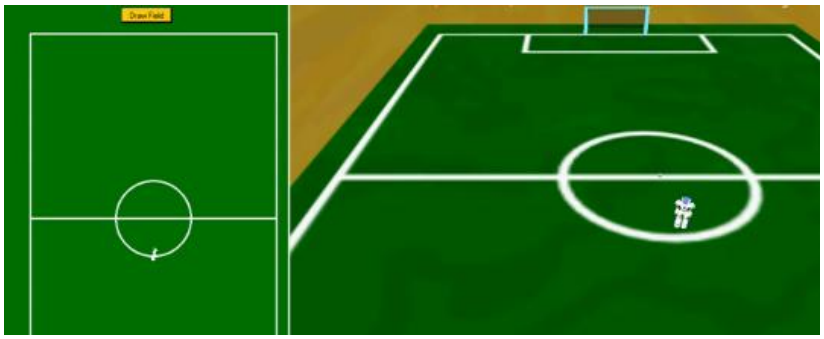
4.2 Forward, Side and Angular Kicks

All kick related behaviors are generated by a detailed analysis and tuning of the joints on the simulator. There are different types of kicks that are implemented: (a) side kick, (b) angular kick and (c) forward kick (or shoot). The side kick is implemented to pass the ball to a support player. The angular kick is designed for dribbling and for passing the ball to another teammate at different angles. The forward kick or shoot is designed for scoring a goal as well as for long distance passes. Left and right versions of these kicks are available. The selection of leg for the kick/pass (right or left) is determined by the angles provided to the agent. A change of sign determines the preferred leg to be used.

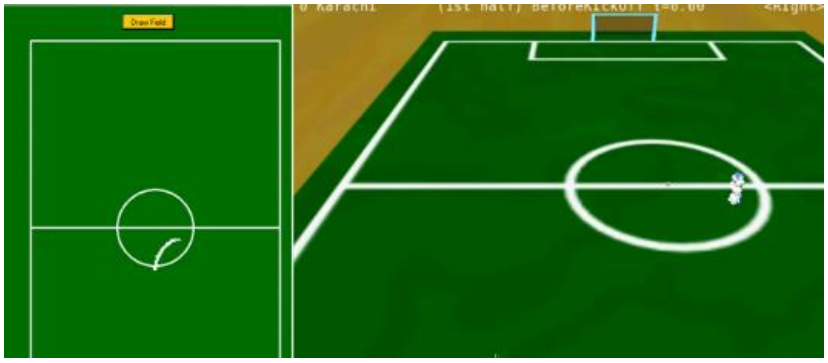
5 Localization

Our localization module currently uses a particle filter and works in the following manner. If more than one marker is available through preceptor/vision sensors then it uses the triangulation method to compute the location and orientation of the corresponding player. It also reinitializes/resamples particles within the neighborhood of the player's position and

orientation in this step. If only one marker is available, then first it predicts the position of each particle using the motion equation and second updates the position of the player using the weighted average of particles' coordinates. Particles whose coordinates and orientation are more consistent with the available preceptor information are assigned more weight. In case no marker is available then the motion equation is applied to each particle and the position of the player is updated as a weighted average of particles' coordinates



(a)



(b)

Fig. 3. Particle Filter Testing Module

During all walk routines (except when the player has fallen down or is kicking the ball), the player keeps rotating its head to make sure that at least one flag is within its vision to better localize itself. In addition, a player who sees the ball and is also confident of its own position, that is, is seeing at least two markers, announces the ball position through the message passing mechanism of the RoboCup 3D Server. This helps the other team members, who are not seeing the ball, in their decision-making. For instance, a defender who has gone too far and is not seeing the ball directly can receive the message from the goal keeper that the ball is behind and it can use back walk to get closer to the ball.

A supported visualization module has also been written to check the consistency/accuracy of the particle filter implementation. The module estimates the position of a player on the field and draws its estimated movement on a separate visualization window to provide a visual verification of the implementation. Fig. 3 shows a walking Nao and how its position and orientation is traced through the particle filter algorithm in the accompanying window on the left. After the release of the Roboviz library, however, we have stopped further development of our own localization support module and have been using RoboViz extensively. In addition to dynamic placement of agents and ball anywhere in the field, RoboViz allows one to draw several figures/annotations next to any team player highlighting its orientation, internal states/beliefs, message communication, etc. These features are extremely helpful in fine-tuning localization and strategy routines

6 Strategy

Being a new team, a significant portion of our efforts to date have been focused on developing competitive locomotion and localization mechanisms. As of now, our strategy module is capable of team formation, role switching, adversary/situation awareness and priority-based fuzzy rules. Four different types of roles are defined for the players: goal keeper, defenders, attacker and supporters. Defenders are further broken down into left and right defenders depending upon their default positions. The attacker/supporters role is dynamic and any player in this set can become an attacker if it satisfies certain conditions. The remaining players then take the supporter role automatically. State machines in the form of priority fuzzy rules are defined for each role. These rules take inputs from a player's vision and the messages it receives from other team members. The rules are defined after careful analysis of several training matches. Our current focus is on enhancing the existing strategy via application of machine learning techniques on game logs. Similarly, incorporation of case-based and inductive reasoning is another area of active research and development. We are also implementing path planning and collision avoidance algorithms as part of our team strategy. In addition, we aim to apply our work on strategy optimization in dynamic uncertain situation within the realm of RoboCup Soccer [17- 21].

Acknowledgement

We are extremely grateful to Drew Noakes, the creator of TinMan, and to Justin Stocker, the creator of RoboViz, for their continuous and prompt support that has immensely helped us in gaining a better understanding of the respective libraries.

References

1. Anshar, M and Williams, M-A: Extended Evolutionary Fast Learn-to-Walk Approach for Four-Legged Robots. Journal of Bionic Engineering, 255-263 (2007).
2. Anshar, M and Williams, M-A: Evolutionary Robot Gaits. In: International Conference on Intelligent Unmanned Systems(2007)

3. Karol, A. and Williams, M-A.: Robot soccer Distributed Sensor Fusion for Object Tracking. In: RoboCup 2005, Robot Soccer World Cup VIII Series: Lecture Notes in Computer Science, Lecture Notes in Artificial Intelligence, 504-511 (2006)
4. Stanton, C. and Williams, M-A.: A Novel and Practical Approach towards Color Constancy Mobile Robots. In: RoboCup 2005: Robot Soccer World Cup VIII Series: Lecture Notes in Computer Science, Lecture Notes in AI, 444-451 (2006)
5. Stanton, C. and Williams, M-A.: Grounding Robot Sensory and Symbolic Information using the Semantic Web. In: RoboCup 2003: Robot Soccer World Cup VII Series: Lecture Notes in Computer Science, Lecture Notes in Artificial Intelligence, Vol. 3020 Polani, D.; Browning, B.; Bonarini, A.; Yoshida, K. (Eds.) 757-765 (2004)
6. Karol, A., Nebel, B., and Williams, M-A.: Case-Based Game Play in the RoboCup Four-Legged League: Part I The Theoretical Model. In: RoboCup 2003: Robot Soccer World Cup VII Series: Lecture Notes in Computer Science,: Lecture Notes in Artificial Intelligence, Vol. 3020 Polani, D.; Browning, B.; Bonarini, A.; Yoshida, K. (Eds.) 739 -748 (2004)
7. Gärdenfors, P. and Williams, M-A, Building Rich and Grounded Robot World Models from Sensors and Knowledge Resources: A Conceptual Spaces Approach. In:2nd International Symposium on Autonomous Mini-Robots for Research and Edutainment, 34-45 (2003)
8. Mendoza, R., Johnston, B., Yang, F., Huang, Z., Chen, X., and Williams, M-A.: OBOC: Ontology based object categorisation for robots. In: Fourth International Conference on Computational Intelligence, Robotics and Autonomous Systems (2007)
9. http://www.wrighteagleunleashed.org
10. http://code.google.com/p/tin-man/
11. https://launchpad.net/littlegreenbats
12. http://sites.google.com/site/zigorat3d/
13. https://sites.google.com/site/umroboviz/
14. Shafii, N., Khorsandian, A., Abolmaleki, A. and Jozi, B.: An Optimized Gait Generator Based on Fourier Series Towards Fast and Robust Biped Locomotion Involving Arms Swing. In: IEEE International Conference on Automation and Logistics (2009)
15. Picado, H., Gestal, M. Lau, N, Reis, L. P. and Tom, A. M.: Automatic Generation of Biped Walk Behavior Using Genetic Algorithms. Lecture Notes in Computer Science, Volume 5517, pp. 805-812 (2009)
16. Heinen, M.R. Osorio, F.S.: Applying Genetic Algorithms to Control Gait of Simulated Robots. In: Electronics, Robotics and Automotive Mechanics Conference, (2007)
17. Haider, S., From Dynamic Influence Nets to Dynamic Bayesian Networks: A Transformation Algorithm, International Journal of Intelligent Systems, 24(2), pp. 919-933 (2009)
18. Haider, S. and Levis, A. H., Finding Effective Coruses of Action using Particle Swarm Intelligence, In Proceedings of IEEE World Congress on Computational Intelligence, Hong Kong, pp. 1135-1140 (2008)
19. Haider, S., Zaidi, A. K. and Levis, A. H, Identification of Best Set of Actions in Influence Nets, International Journal of Hybrid Intelligent Systems (IJHIS), 5 (1), pp. 19-29 (2008)
20. Haider, S. and Levis, A. H, Modeling Time-varying Uncertain Situations using Dynamic Influence Nets,International Journal of Approximate Reasoning, 49 (2), pp. 488-502 (2008)
21. Haider, S. and Levis, A. H, Effective Courses of Action Determination to Achieve Desired Effects, IEEE Transactions on Systems, Man, and Cybernetics - Part A,, 36 (6), pp. 1140-1150 (2007)